Exercise: 5

Name: Pekka Lehtola

How many tasks did you do: 6

Were the tasks easy, ok, difficult: difficult

Do you need help/comments in any task (if yes, to which ones):

Tehtävästä 6 voisi olla tunnilla esimerkki.

Explain the following term and what is it used for:

a. Multiple inheritance

A class can be derived from more than one base class in Python, similar to C++. This is called multiple inheritance. In multiple inheritance, the features of all the base classes are inherited into the derived class. The syntax for multiple inheritance is similar to single inheritance.

Example:

```
class Base1:
    pass

class Base2:
    pass

class MultiDerived(Base1, Base2):
    pass
```

2. True or false?

a. The practice of procedural programming is centered on the creation of objects.

False. The focus of procedural programming is to break down a programming task into a collection of variables, data structures, and subroutines, whereas in object-oriented programming it is to break down a programming task into objects that expose behavior (methods) and data (members or attributes) using interfaces.

b. Object reusability has been a factor in the increased use of object-oriented programming.

True. OOP was developed to increase the reusability and maintainability of source code. This is the reason its used in bigger projects.

c. It is a common practice in object-oriented programming to make all of a class's data attributes accessible to statements outside the class.

   False. Most of the time Setters and Getters are used to modify data attributes, for security reasons.

d. A class methods does not have to have a self parameter.

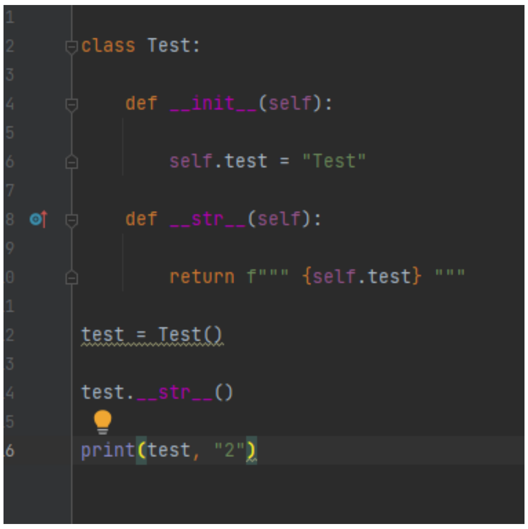
   False/True Self can be any other word, but usually "self", but that word is mandatory.

e. Starting an attribute name with two underscores will hide the attribute from code outside the class.

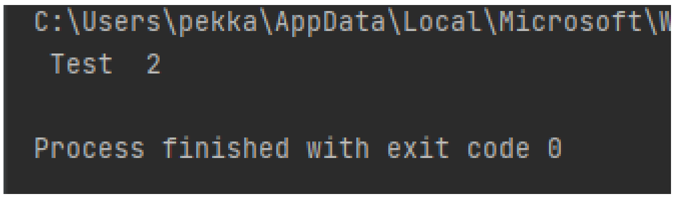
   True. Double underscore prefix prevents access to attribute, except through accessors.

f. You cannot directly call the __str__method.

   True



```python

class Test:

    def __init__(self):

        self.test = "Test"

    def __str__(self):

        return f""" {self.test} """

test = Test()

test.__str__()

print(test, "2")
```



```
C:\Users\pekka\AppData\Local\Microsoft\W
 Test  2

Process finished with exit code 0
```

3. Multiple choice:

a. The ______ method is automatically called when an object is created.

i. __init__

ii. init

iii. __str__

iv. __object__

b. The ______ programming practice is centered on creating functions that are separated from the data that they work on.

i. modular

ii. procedural

iii. functional

iv. object-oriented

c. The ______ programming practice is centered on creating objects.

i. object-centric

ii. objective

iii. procedural

iv. object-oriented

d. A(n) ______ is a component of a class that references data

i. method

ii. instance

iii. data attribute

iv. module

e. By doing this, you can hide a class's attribute from code outside the class.

i. avoiding using the self-parameter to create the attribute

ii. begin the attribute's name with private__

iii. begin the name of the attribute with two underscores

iv. begin the name of the attribute with the symbol #

f. A(n) ______ method stores a value in the data attribute or changes its value in some other way.

i. modifier

ii. constructor

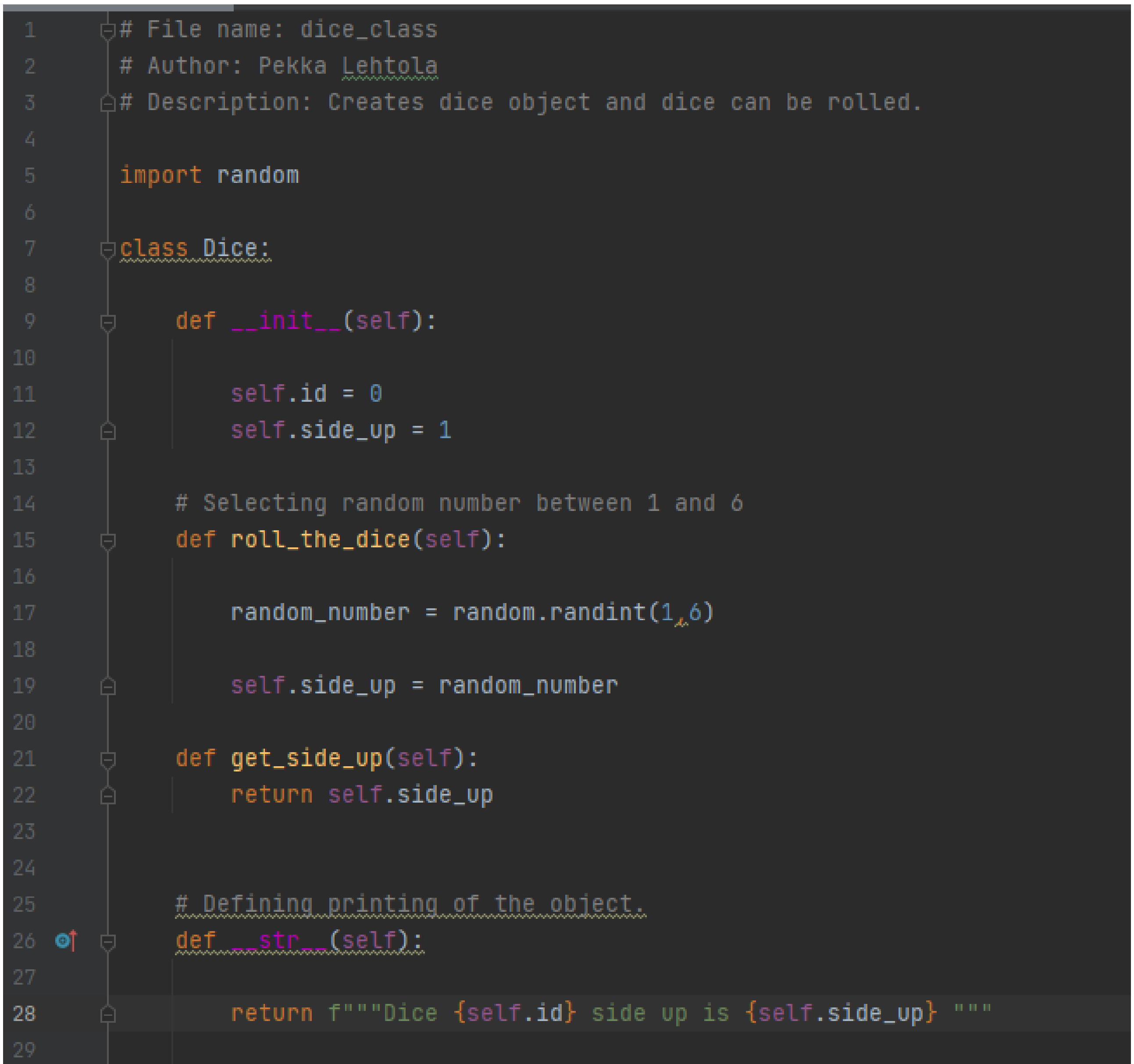
iii. mutator

iv. Accessor

4. Create multiple dices (at least three) and put them in a list. See that your dice can be rolled and the side can be shown. Create a small game where the best sum of three rolls wins. If the sum is a tie, tied dices are rolled as long as a winner is found (best side wins). Use functions and pass objects (or list of objects) as arguments. Use informative and clear output prints.

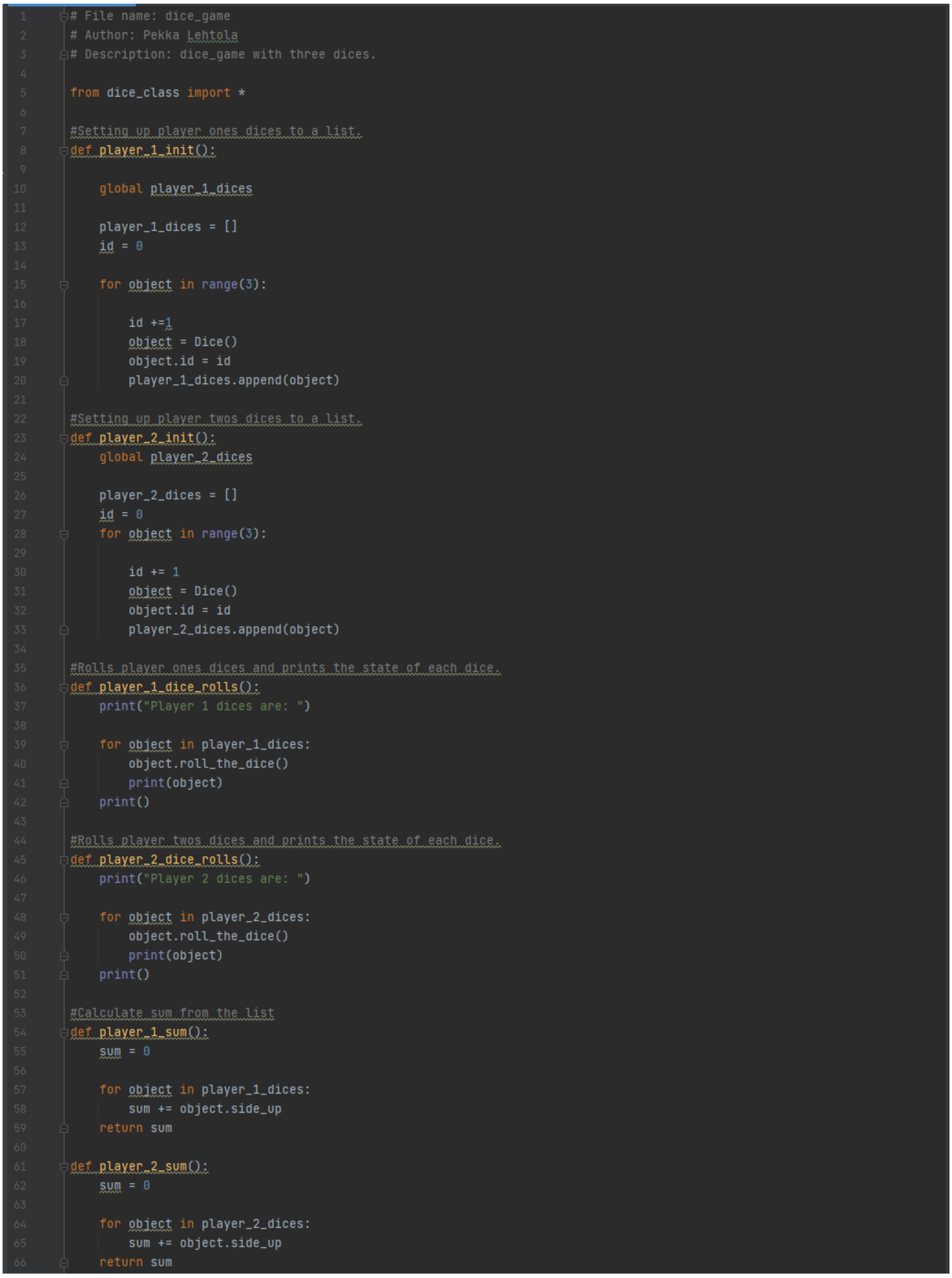
Screen capture of Task 4

Dice class:

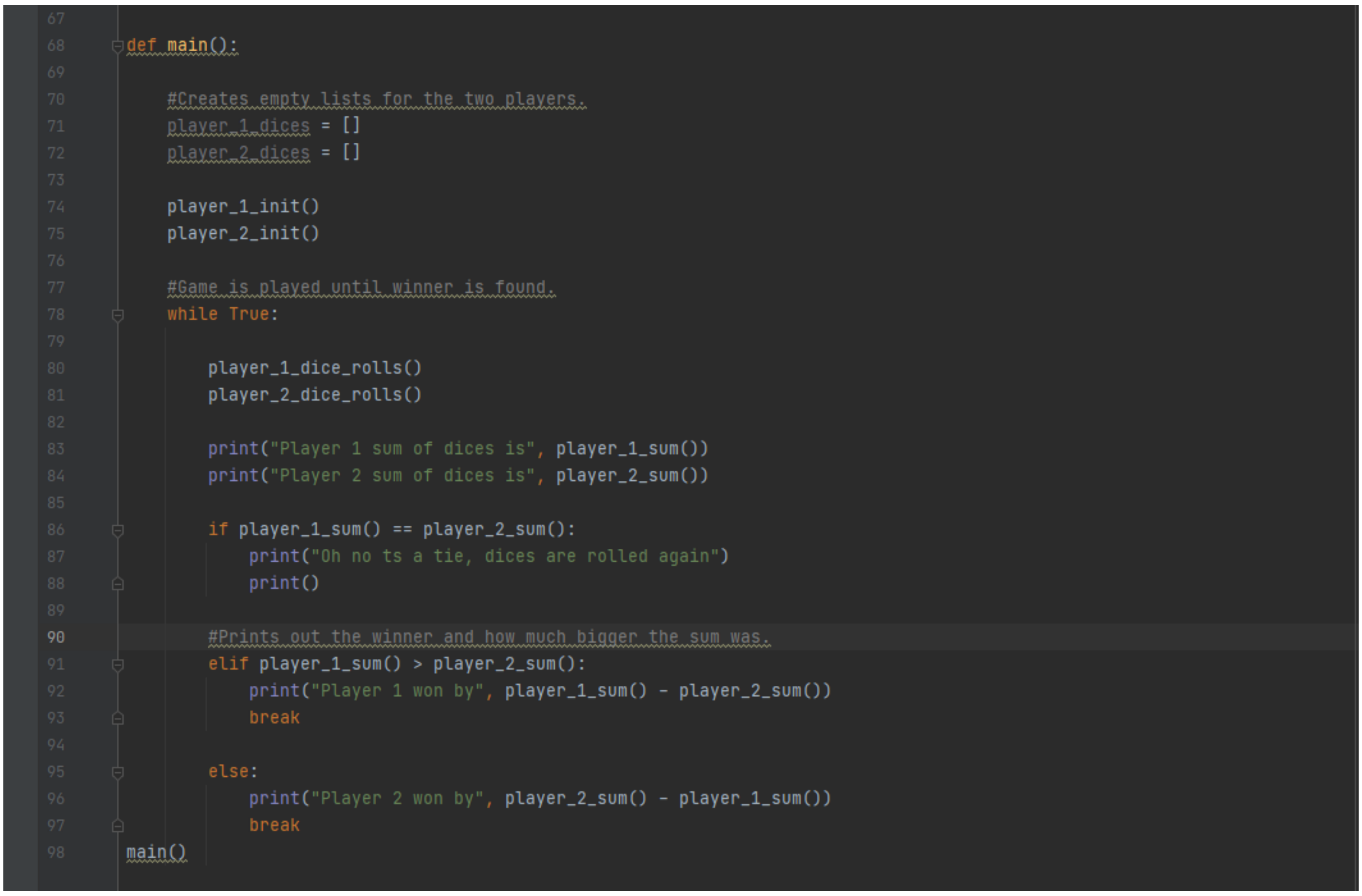


```python
# File name: dice_class
# Author: Pekka Lehtola
# Description: Creates dice object and dice can be rolled.

import random

class Dice:

    def __init__(self):

        self.id = 0
        self.side_up = 1

    # Selecting random number between 1 and 6
    def roll_the_dice(self):

        random_number = random.randint(1,6)

        self.side_up = random_number

    def get_side_up(self):
        return self.side_up


    # Defining printing of the object.
    def __str__(self):

        return f"""Dice {self.id} side up is {self.side_up} """

```

Main game:

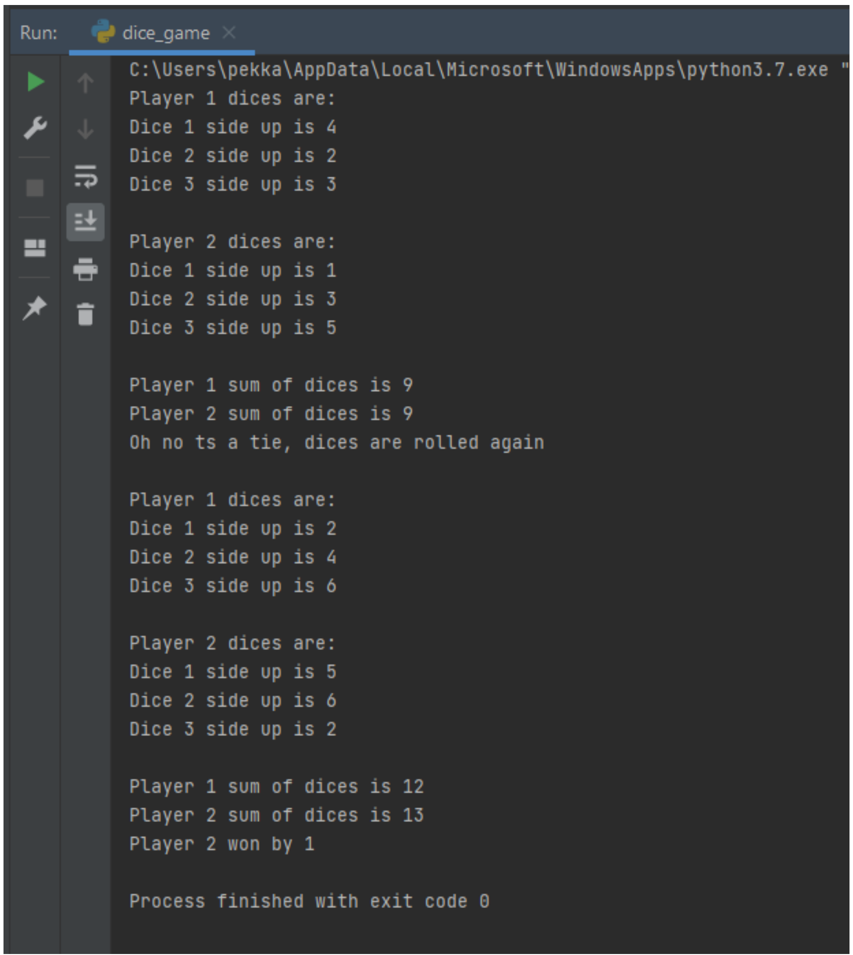


```python
# File name: dice_game
# Author: Pekka Lehtola
# Description: dice_game with three dices.

from dice_class import *

#Setting up player ones dices to a list.
def player_1_init():

    global player_1_dices

    player_1_dices = []
    id = 0

    for object in range(3):

        id +=1
        object = Dice()
        object.id = id
        player_1_dices.append(object)

#Setting up player twos dices to a list.
def player_2_init():
    global player_2_dices

    player_2_dices = []
    id = 0
    for object in range(3):

        id += 1
        object = Dice()
        object.id = id
        player_2_dices.append(object)

#Rolls player ones dices and prints the state of each dice.
def player_1_dice_rolls():
    print("Player 1 dices are: ")

    for object in player_1_dices:
        object.roll_the_dice()
        print(object)
    print()

#Rolls player twos dices and prints the state of each dice.
def player_2_dice_rolls():
    print("Player 2 dices are: ")

    for object in player_2_dices:
        object.roll_the_dice()
        print(object)
    print()

#Calculate sum from the list
def player_1_sum():
    sum = 0

    for object in player_1_dices:
        sum += object.side_up
    return sum

def player_2_sum():
    sum = 0

    for object in player_2_dices:
        sum += object.side_up
    return sum
```



```python

def main():

    #Creates empty lists for the two players.
    player_1_dices = []
    player_2_dices = []

    player_1_init()
    player_2_init()

    #Game is played until winner is found.
    while True:

        player_1_dice_rolls()
        player_2_dice_rolls()

        print("Player 1 sum of dices is", player_1_sum())
        print("Player 2 sum of dices is", player_2_sum())

        if player_1_sum() == player_2_sum():
            print("Oh no ts a tie, dices are rolled again")
            print()

        #Prints out the winner and how much bigger the sum was.
        elif player_1_sum() > player_2_sum():
            print("Player 1 won by", player_1_sum() - player_2_sum())
            break

        else:
            print("Player 2 won by", player_2_sum() - player_1_sum())
            break
main()
```

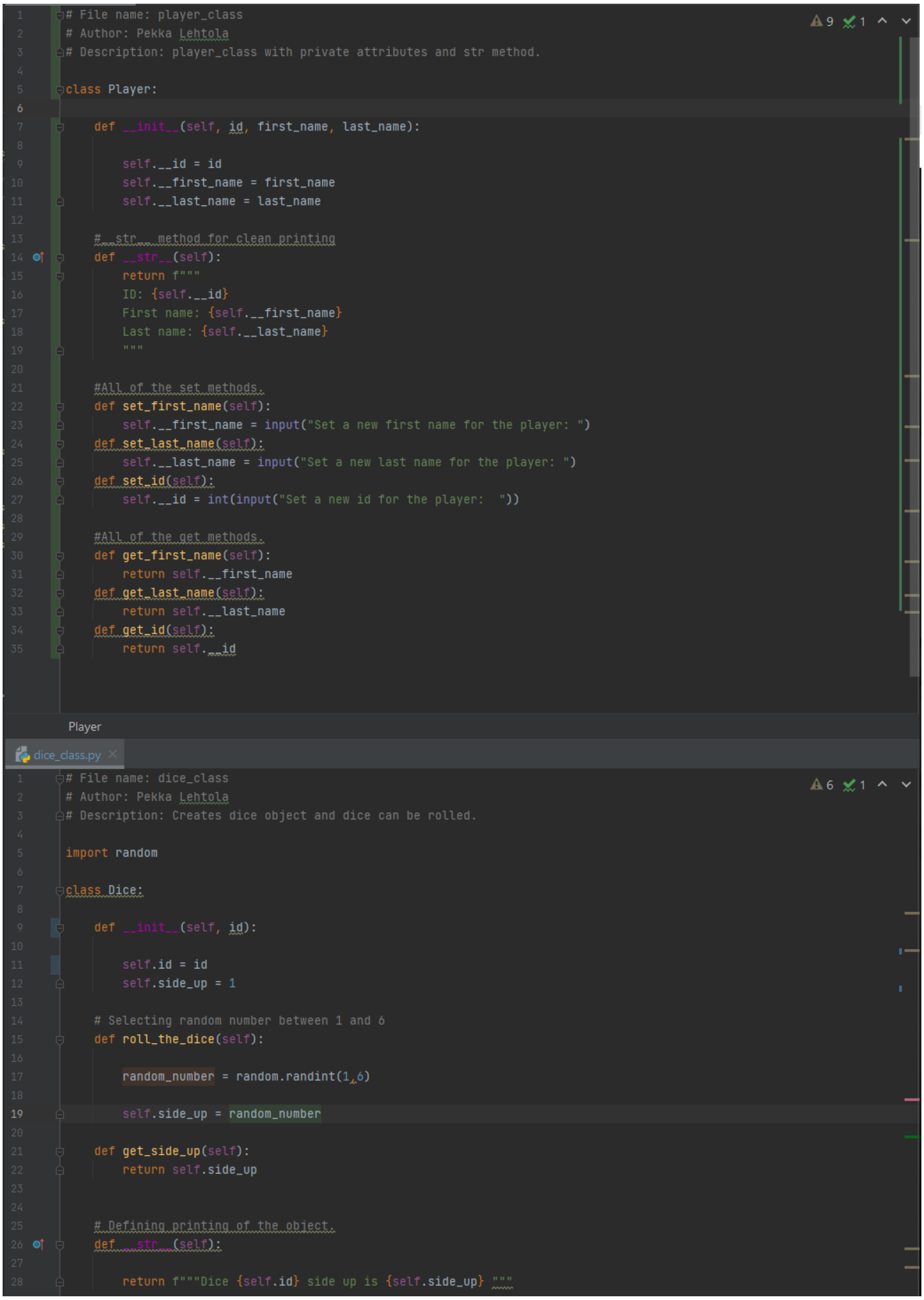
Screen capture of the output of Task 4



```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe "
Player 1 dices are:
Dice 1 side up is 4
Dice 2 side up is 2
Dice 3 side up is 3

Player 2 dices are:
Dice 1 side up is 1
Dice 2 side up is 3
Dice 3 side up is 5

Player 1 sum of dices is 9
Player 2 sum of dices is 9
Oh no ts a tie, dices are rolled again

Player 1 dices are:
Dice 1 side up is 2
Dice 2 side up is 4
Dice 3 side up is 6

Player 2 dices are:
Dice 1 side up is 5
Dice 2 side up is 6
Dice 3 side up is 2

Player 1 sum of dices is 12
Player 2 sum of dices is 13
Player 2 won by 1

Process finished with exit code 0
```

5. Create a class called Player. Player has at least the following data attributes: first name, last name and a player id. Remember to code accessor and mutator methods and strmethod. Create a dictionary so that the player id is a key and each player has one dice. Roll each player's dice and print out each player's dice's side. Use informative and clear output prints.

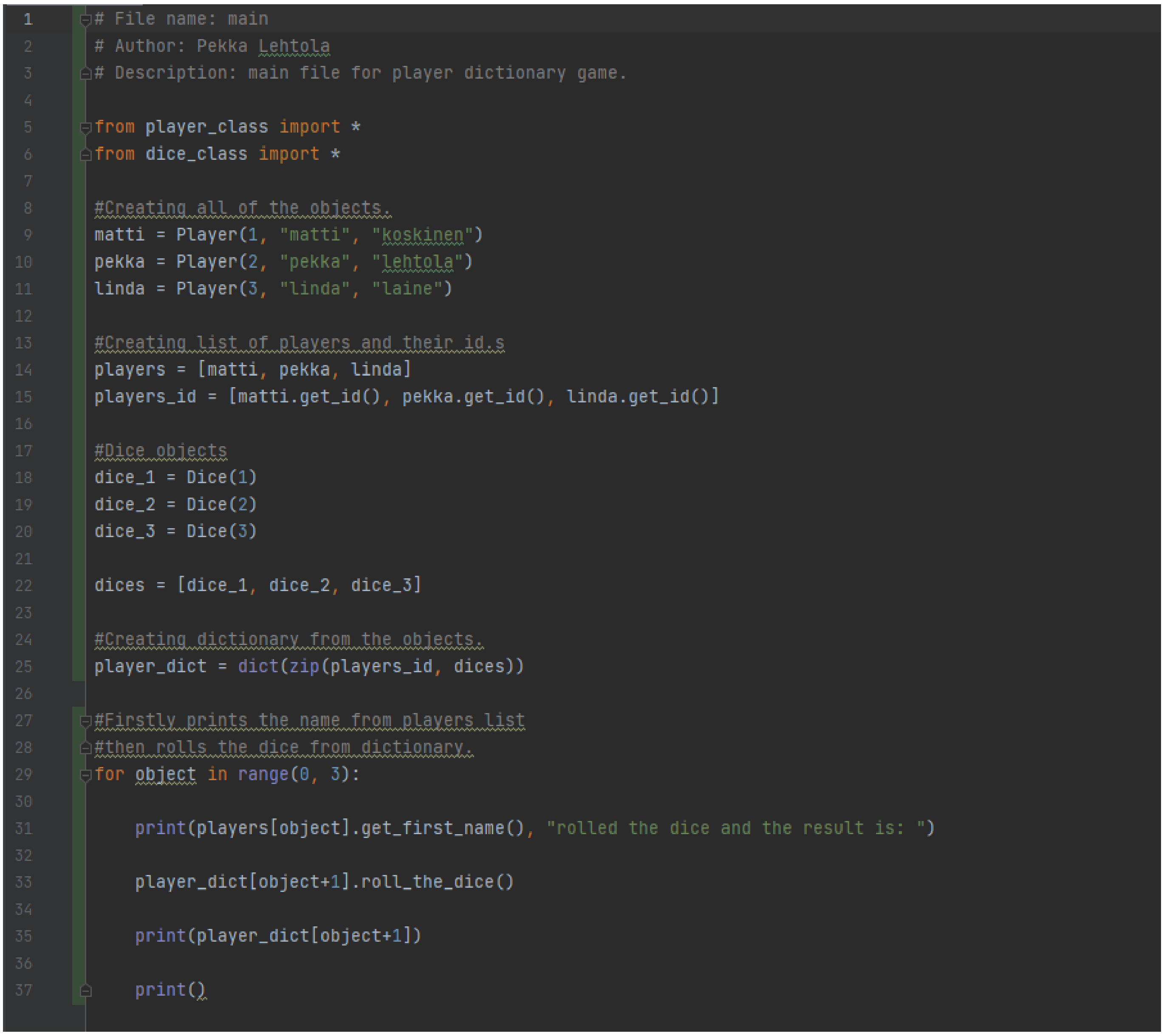
Screen capture of Task 5



```python
# File name: player_class
# Author: Pekka Lehtola
# Description: player_class with private attributes and str method.

class Player:

    def __init__(self, id, first_name, last_name):

        self.__id = id
        self.__first_name = first_name
        self.__last_name = last_name

    # __str__ method for clean printing
    def __str__(self):
        return f"""
        ID: {self.__id}
        First name: {self.__first_name}
        Last name: {self.__last_name}
        """

    #All of the set methods.
    def set_first_name(self):
        self.__first_name = input("Set a new first name for the player: ")
    def set_last_name(self):
        self.__last_name = input("Set a new last name for the player: ")
    def set_id(self):
        self.__id = int(input("Set a new id for the player:  "))

    #All of the get methods.
    def get_first_name(self):
        return self.__first_name
    def get_last_name(self):
        return self.__last_name
    def get_id(self):
        return self.__id
```

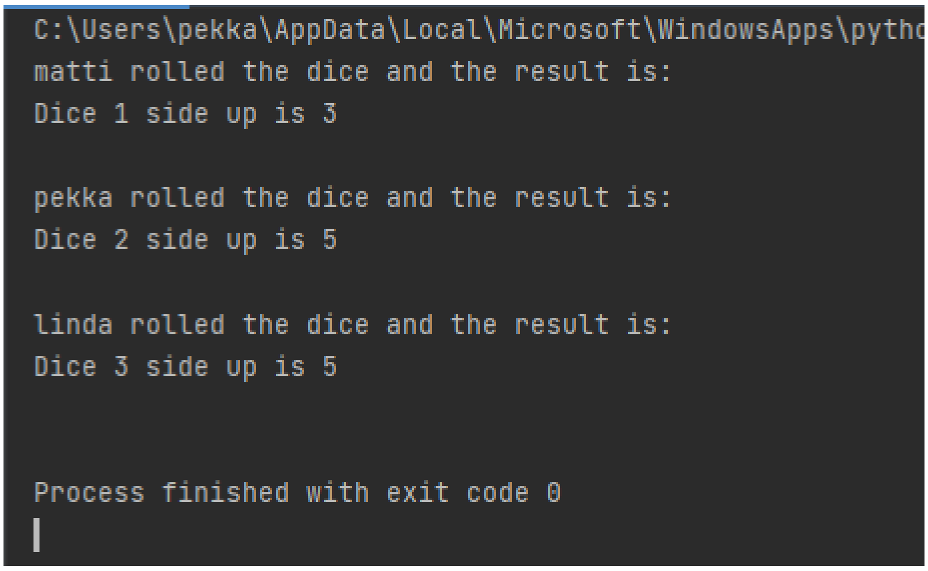
```python
# File name: dice_class
# Author: Pekka Lehtola
# Description: Creates dice object and dice can be rolled.

import random

class Dice:

    def __init__(self, id):

        self.id = id
        self.side_up = 1

    # Selecting random number between 1 and 6
    def roll_the_dice(self):

        random_number = random.randint(1,6)

        self.side_up = random_number

    def get_side_up(self):
        return self.side_up


    # Defining printing of the object.
    def __str__(self):

        return f"""Dice {self.id} side up is {self.side_up} """
```

Main:



```python
# File name: main
# Author: Pekka Lehtola
# Description: main file for player dictionary game.

from player_class import *
from dice_class import *

#Creating all of the objects.
matti = Player(1, "matti", "koskinen")
pekka = Player(2, "pekka", "lehtola")
linda = Player(3, "linda", "laine")

#Creating list of players and their id.s
players = [matti, pekka, linda]
players_id = [matti.get_id(), pekka.get_id(), linda.get_id()]

#Dice objects
dice_1 = Dice(1)
dice_2 = Dice(2)
dice_3 = Dice(3)

dices = [dice_1, dice_2, dice_3]

#Creating dictionary from the objects.
player_dict = dict(zip(players_id, dices))

#Firstly prints the name from players list
#then rolls the dice from dictionary.
for object in range(0, 3):

    print(players[object].get_first_name(), "rolled the dice and the result is: ")

    player_dict[object+1].roll_the_dice()

    print(player_dict[object+1])

    print()
```

Screen capture of the output of Task 5

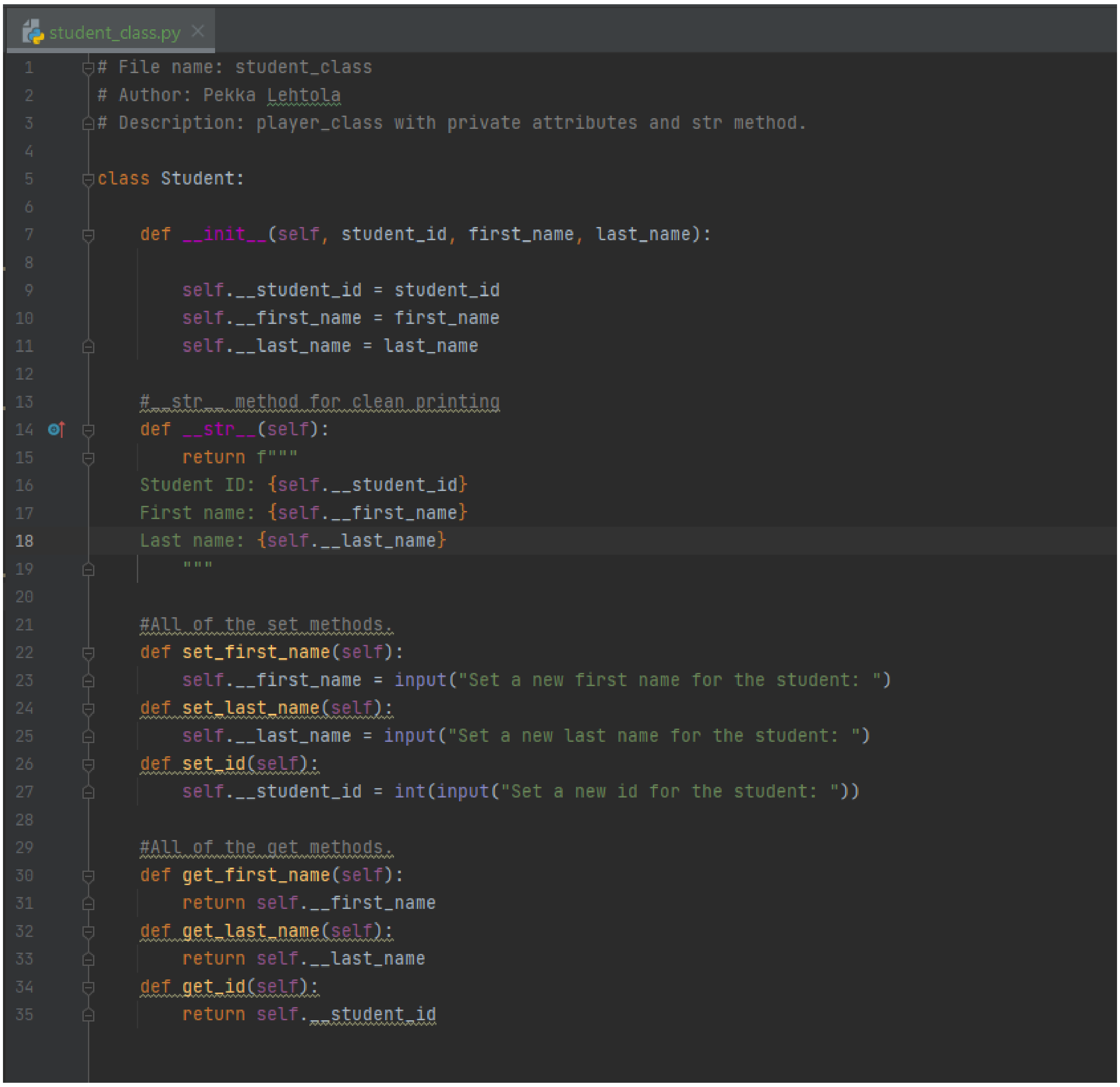


```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\pytho
matti rolled the dice and the result is:
Dice 1 side up is 3

pekka rolled the dice and the result is:
Dice 2 side up is 5

linda rolled the dice and the result is:
Dice 3 side up is 5


Process finished with exit code 0
```

6. Create a class called Student and use the following data attributes: first name, last name and student id. Remember to code accessor and mutator methods and str-method. Store students and their pet mammal to dictionary (use the mammals from Exercise 4). Think, what should be used as the dictionary key. Code a function that prints out each student and their mammal's information. Use informative and clear output print.

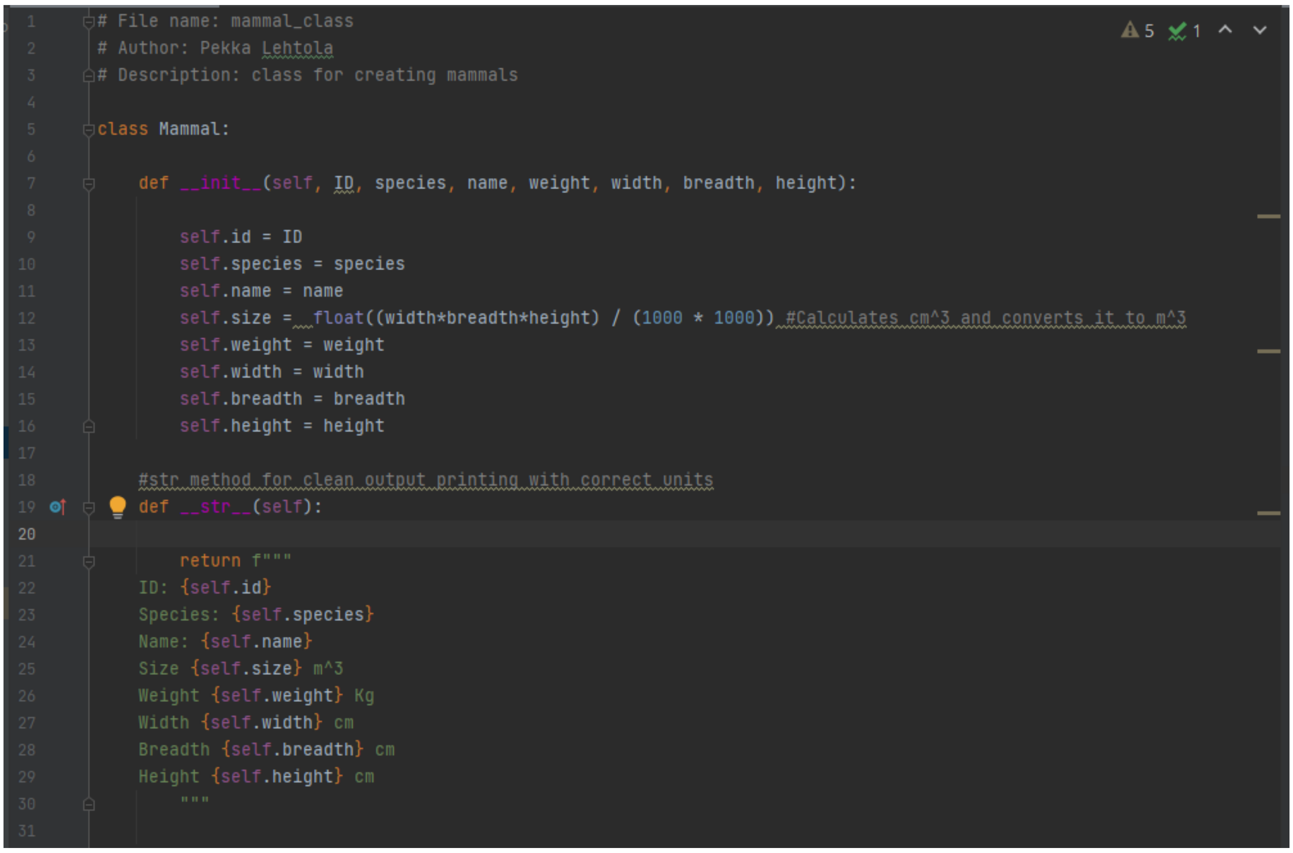
Screen capture of Task 6

Student class:


student_class.py

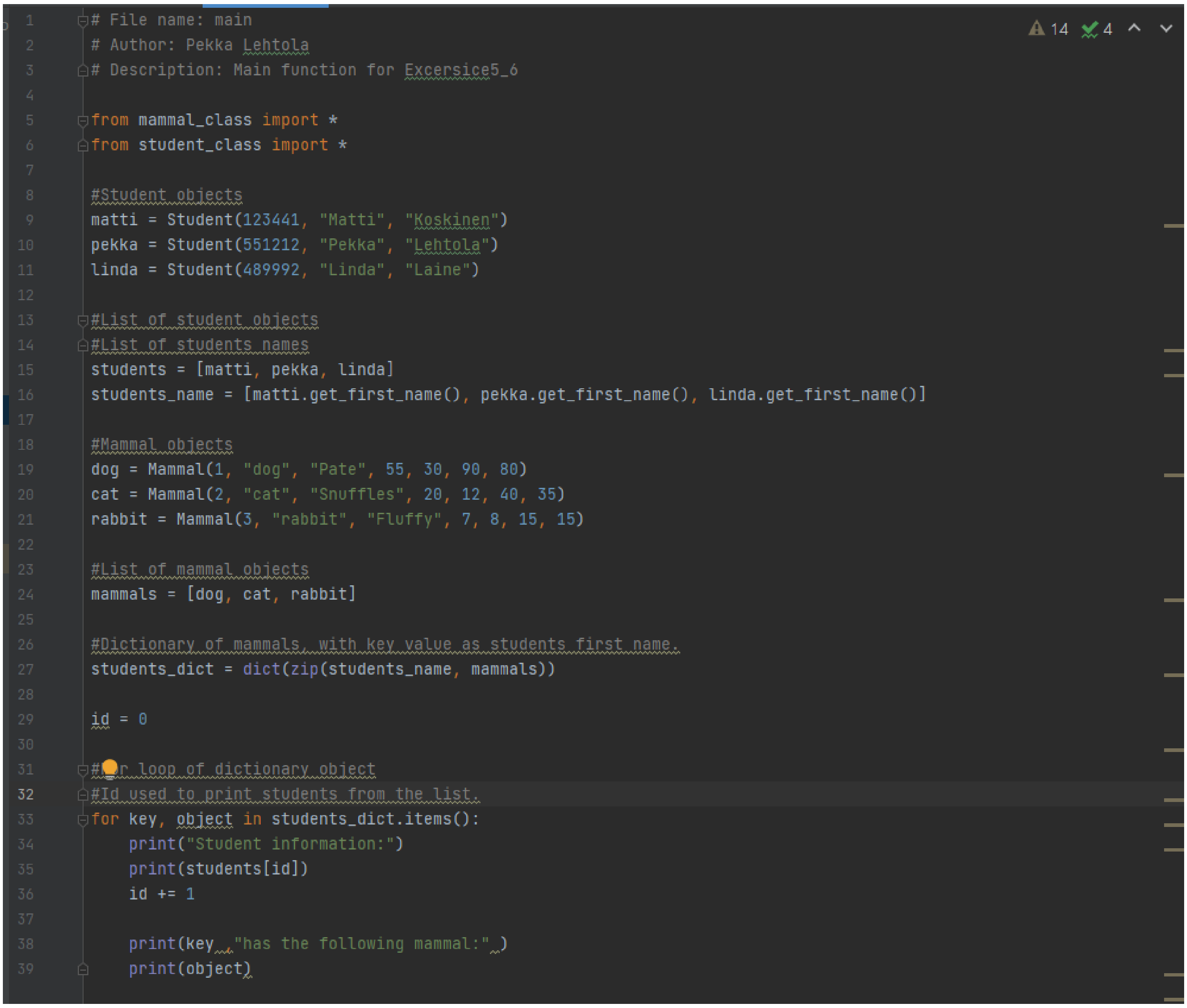
```python
# File name: student_class
# Author: Pekka Lehtola
# Description: player_class with private attributes and str method.

class Student:

    def __init__(self, student_id, first_name, last_name):

        self.__student_id = student_id
        self.__first_name = first_name
        self.__last_name = last_name

    #__str__ method for clean printing
    def __str__(self):
        return f"""
    Student ID: {self.__student_id}
    First name: {self.__first_name}
    Last name: {self.__last_name}
        """

    #All of the set methods.
    def set_first_name(self):
        self.__first_name = input("Set a new first name for the student: ")
    def set_last_name(self):
        self.__last_name = input("Set a new last name for the student: ")
    def set_id(self):
        self.__student_id = int(input("Set a new id for the student: "))

    #All of the get methods.
    def get_first_name(self):
        return self.__first_name
    def get_last_name(self):
        return self.__last_name
    def get_id(self):
        return self.__student_id
```

Mammal class:



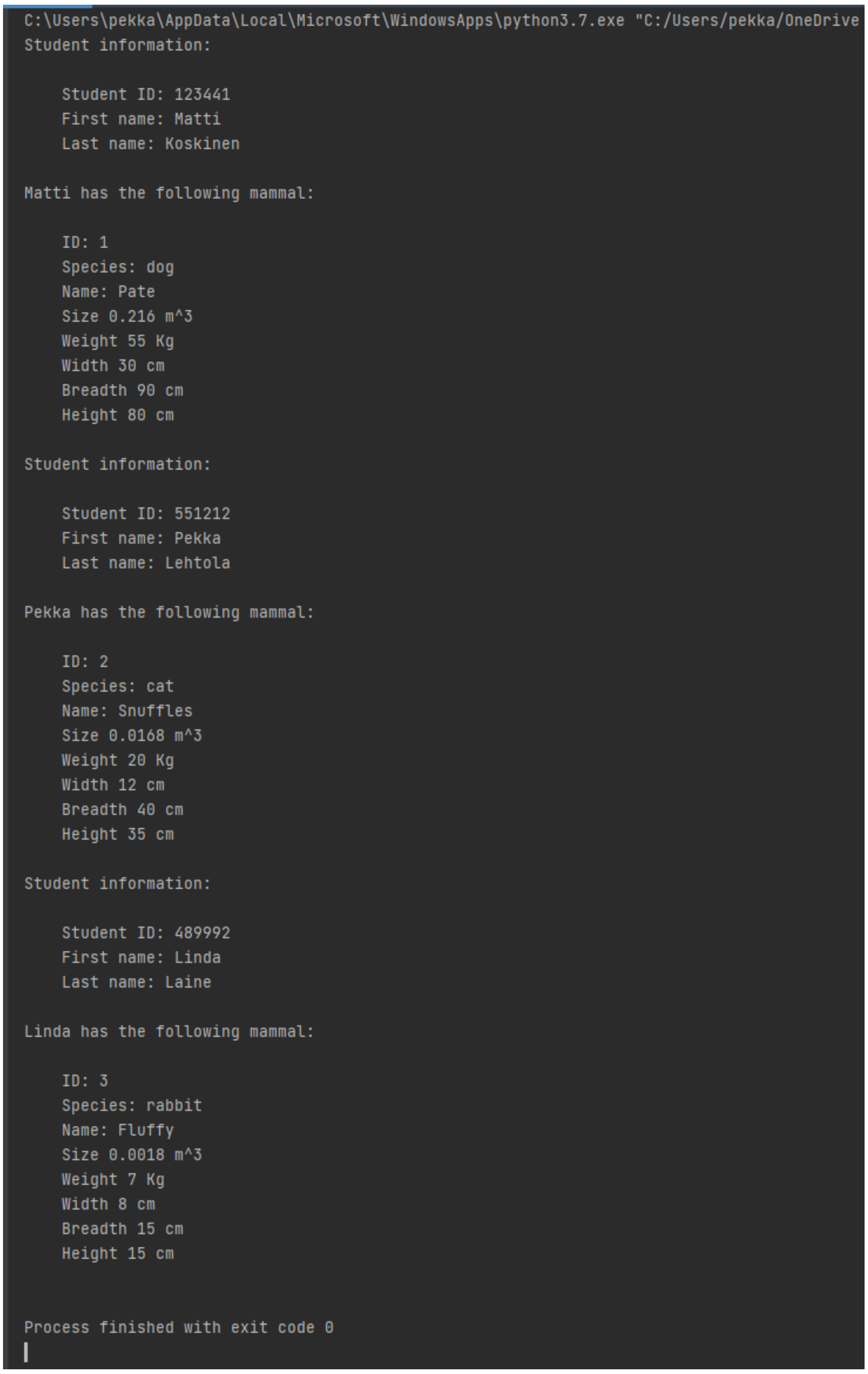
```
# File name: mammal_class
# Author: Pekka Lehtola
# Description: class for creating mammals

class Mammal:

    def __init__(self, ID, species, name, weight, width, breadth, height):

        self.id = ID
        self.species = species
        self.name = name
        self.size = float((width*breadth*height) / (1000 * 1000)) #Calculates cm^3 and converts it to m^3
        self.weight = weight
        self.width = width
        self.breadth = breadth
        self.height = height

    #str method for clean output printing with correct units
    def __str__(self):

        return f"""
    ID: {self.id}
    Species: {self.species}
    Name: {self.name}
    Size {self.size} m^3
    Weight {self.weight} Kg
    Width {self.width} cm
    Breadth {self.breadth} cm
    Height {self.height} cm
        """

```

Main:



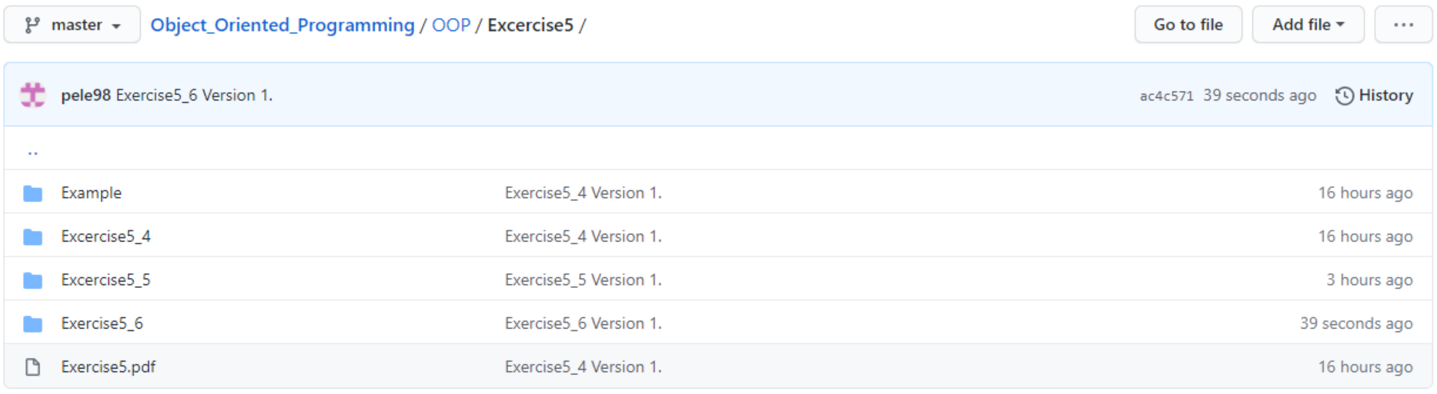
```python
# File name: main
# Author: Pekka Lehtola
# Description: Main function for Excersice5_6

from mammal_class import *
from student_class import *

#Student objects
matti = Student(123441, "Matti", "Koskinen")
pekka = Student(551212, "Pekka", "Lehtola")
linda = Student(489992, "Linda", "Laine")

#List of student objects
#List of students names
students = [matti, pekka, linda]
students_name = [matti.get_first_name(), pekka.get_first_name(), linda.get_first_name()]

#Mammal objects
dog = Mammal(1, "dog", "Pate", 55, 30, 90, 80)
cat = Mammal(2, "cat", "Snuffles", 20, 12, 40, 35)
rabbit = Mammal(3, "rabbit", "Fluffy", 7, 8, 15, 15)

#List of mammal objects
mammals = [dog, cat, rabbit]

#Dictionary of mammals, with key value as students first name.
students_dict = dict(zip(students_name, mammals))

id = 0

#r loop of dictionary object
#Id used to print students from the list.
for key, object in students_dict.items():
    print("Student information:")
    print(students[id])
    id += 1

    print(key,"has the following mammal:",)
    print(object)
```

Screen capture of the output of Task 6



```
C:\Users\pekka\AppData\Local\Microsoft\WindowsApps\python3.7.exe "C:/Users/pekka/OneDrive
Student information:

    Student ID: 123441
    First name: Matti
    Last name: Koskinen

Matti has the following mammal:

    ID: 1
    Species: dog
    Name: Pate
    Size 0.216 m^3
    Weight 55 Kg
    Width 30 cm
    Breadth 90 cm
    Height 80 cm

Student information:

    Student ID: 551212
    First name: Pekka
    Last name: Lehtola

Pekka has the following mammal:

    ID: 2
    Species: cat
    Name: Snuffles
    Size 0.0168 m^3
    Weight 20 Kg
    Width 12 cm
    Breadth 40 cm
    Height 35 cm

Student information:

    Student ID: 489992
    First name: Linda
    Last name: Laine

Linda has the following mammal:

    ID: 3
    Species: rabbit
    Name: Fluffy
    Size 0.0018 m^3
    Weight 7 Kg
    Width 8 cm
    Breadth 15 cm
    Height 15 cm


Process finished with exit code 0
```

Screen capture of git log (showing that you made a commit after every task).



master / Object_Oriented_Programming / OOP / Excercise5 /

Go to file | Add file | ...

pele98 Exercise5_6 Version 1. ac4c571 39 seconds ago History

..		
Example	Exercise5_4 Version 1.	16 hours ago
Excercise5_4	Exercise5_4 Version 1.	16 hours ago
Excercise5_5	Exercise5_5 Version 1.	3 hours ago
Exercise5_6	Exercise5_6 Version 1.	39 seconds ago
Exercise5.pdf	Exercise5_4 Version 1.	16 hours ago

Self-assessment:

This exercise was easy/difficult/ok/etc. for me because…

Tehtävät tuntuivat haastavilta. Sain dictionary objectit hädin tuskin toimimaan.

Doing this exercise, I learned…

Alkeellisen tavan käsitellä objecteja dictionary muodossa.

I am still wondering…

Voiko dictionaryn avain olla myös jotenkin objecti?

I understood/did not understand that… ; I did/did not know that… ; I did/did not manage to do…

Omasta mielestä koodi ei ollut tällä kertaa onnistunutta. Yritin katsoa esimerkkiä, joka löytyi Itsistä, mutten saanut mitään toimimaan.